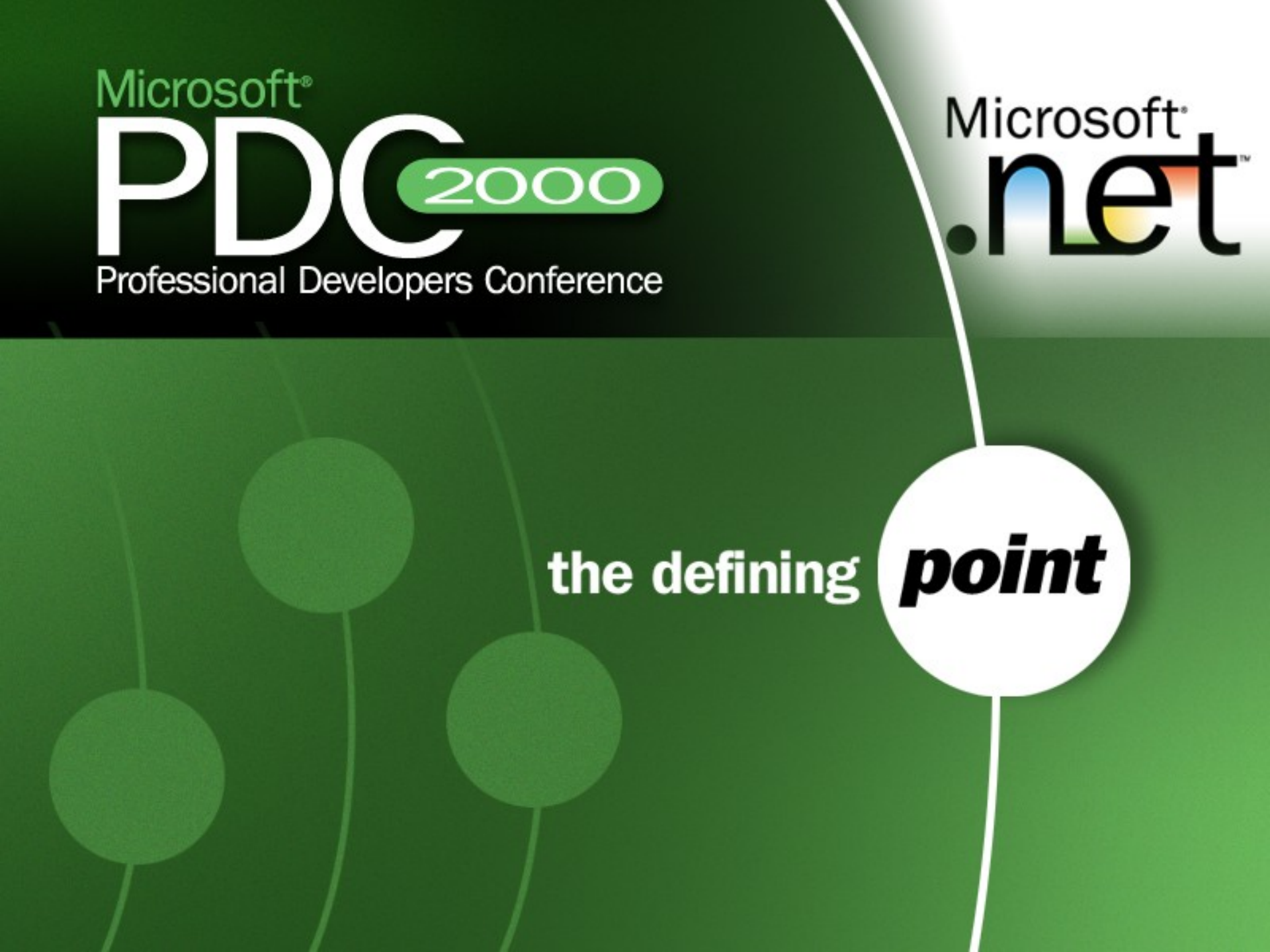
Handling The Basics: Getting The Most Out Of The Frameworks

Jay Roxe
Development Lead
.NET Frameworks
Microsoft Corporation

3-222


Microsoft®
PDC 2000
Professional Developers Conference
Microsoft®
.net™
the defining point

Introduction

- “What are all of these classes anyway?”
  - Wide range of functionality
- Extensibility
- Performance concerns
- Build a small demo application

Agenda

- Introduction
- Base Data Types
- Strings
- Regular Expressions
- Collections
- Globalization
- IO
- Serialization
- Cryptography

Demo
Acme Corp Address Book

Agenda

- Introduction
- Base Data Types
  System
- Strings
- Regular Expressions
- Collections
- Globalization
- IO
- Serialization
- Cryptography

Base Data Types

- Primitive and common types
  - Byte, Int16, Decimal, DateTime, String, etc.
- Object form of primitives
- Formatting and Parsing Functionality
  - ToString/FromString vs. Format/Parse
- Conversion to other types
  - Convert

Agenda

- Introduction
- Base Data Types
- Strings
  System
- Regular Expressions
- Collections
- Globalization
- IO
- Serialization
- Cryptography

String Comparison

- Ordinal versus culturally-aware comparisons
  - CompareOrdinal and Equals are ordinal
  - Compare is culturally-aware
- Convenience wrappers for globalization functionality

String Creation

- Concatenation is the most common
  - String.Concat(“a”, “b”, “c”)
  - “a” + “b” + “c”
- Strings are immutable
- Use a StringBuilder for incremental creation
- Transformation functions return a new string (e.g., ToUpper, ToLower, Trim)

String Formatting

- Format strings for output and display to users
  - Control format, width, padding, alignment
- Format determined with picture strings or codes
  - `String.Format("Please order {0} widgets at {1} each.", i, f)`
  - `String.Format("{0:##00.00}", f)`
  - `String.Format("{0:U}", DateTime.Now)`
- IFormattable provides user control

Agenda

- Introduction
- Base Data Types
- Strings
- Regular Expressions
  System.Text.RegularExpressions
- Collections
- Globalization
- IO
- Serialization
- Cryptography

Regular Expressions

- Provides match and replace capability
- Perl5 and Friedl Compatible
- Match

```
Regex.IsMatch(“Acme Corporation”,”.*e.*n”)
```

- Replace

```
Regex.Replace(“a=b”, “(.*)=(.*)”, “\\2=\\1”)
```

Regular Expressions

- Expression Types
  - Static
  - Interpreted
    - `Regex re = new Regex(“.*e.*n”);`
      `re.IsMatch(“Acme Corporation”);`
  - Compiled
    - `Regex re = new Regex(“.*e.*n”,”c”);`
    - Compiled form is expensive.
      Only use in statics

Agenda

- Introduction
- Base Data Types
- Strings
- Regular Expressions
- Collections
  System.Collections
- Globalization
- IO
- Serialization
- Cryptography

Collections

- Interface Based Model
  - ICollection
    - Enumerate, convert to an array, synchronization, read-only
  - IList, IDictionary
    - Derived from ICollection
    - Add, remove, search
    - IList handles an indexable series of values
    - IDictionary handles a set of key-value pairs

Common Collections

- .NET Frameworks provides some common collections
  - Hashtable, Stack, Queue, ArrayList
- Using an ArrayList (IList)
  - `list[5]="My string data";`
    `String s = (String)list[5];`
- Using a Hashtable (IDictionary)
  - `table[MyDataKey] = MyDataObject;`
    `Data dataObj = (Data)table[MyDataKey];`

Strongly Typed Collections

- Treat generically by interface or specifically by type
- Private interface implementation enables strongly-typed collections

```
public class ButtonList : IList {  public
int Add(Button button)…
int IList.Add(Object obj)…
}
```

Private Interfaces

Enumerate A Collection

- All collections support getting an IEnumerator
  - Enumerators can be strongly typed
- Sample

```
foreach (Button b in MyButtonList) {
      Console.WriteLine(b);
}
```

Agenda

- Introduction
- Base Data Types
- Strings
- Regular Expressions
- Collections
- Globalization
  System.Globalization
- IO
- Serialization
- Cryptography

Globalization: CultureInfo

- Globalization is a design principle of the base classes
- CultureInfo is a set of preferences based on language and culture
- CurrentCulture
  - Date and number formatting
  - String comparison and casing
- CurrentUICulture
  - Resource selection
- Culture can be controlled on a per-thread basis

Globalization

- Provides Unicode support on all platforms
  - Carries copies of required Unicode tables
- Character Type Information
- Text Encodings
  - Convert from Unicode to another format
- Calendar Support

Agenda

- Introduction
- Base Data Types
- Strings
- Regular Expressions
- Collections
- Globalization
- IO
  System.IO
- Serialization
- Cryptography

Streams

- Read and write bytes
  - No knowledge of any specific datatype.
- Stream base class
- Composable streaming model
  - Seamlessly add buffering, piping, cryptography,…
- Synchronous and asynchronous support
  - Implement one or both flavors

Readers And Writers

- Most common point of contact with IO
- Consume streams
- Read and write user types
- Knowledge of stream data format as appropriate.
- .NET Frameworks provide BinaryReader/Writer and StreamReader/Writer.
  - StreamReader/Writer understand encodings

Sample: Writing To A File

```
//Open a FileStream in write mode and wrap it in
    //a StreamWriter.
    StreamWriter writer = File.CreateText
    (“C:\\Temp\\Foo.Bar”);

//Write a String, an int, and a double
    //These will show up as UTF8 encoded text
    //on the Stream
    writer.WriteLine(“Hello, World!”);
    writer.WriteLine(5);
    writer.WriteLine(3.1415926535);

//Close the StreamWriter and the underlying Stream
    writer.Close();
```

Agenda

- Introduction
- Base Data Types
- Strings
- Regular Expressions
- Collections
- Globalization
- IO
- Serialization
  System.Runtime.Serialization
- Cryptography

Serialization

- Stores an object graph to a stream
  for later reinstantiation
- Fully automatic
  - System tracks and restores object data and connections between objects
  - Supports all object types
  - ISerializable allows user control
- Pluggable formatter architecture

Agenda

- Introduction
- Base Data Types
- Strings
- Regular Expressions
- Collections
- Globalization
- IO
- Serialization
- Cryptography
  System.Security.Cryptography

Cryptography

- "This is not your mother's CryptoAPI"
  - Make it easy for most developers to use crypto
  - Major customers include Web application developers
- Simple object model for doing crypto operations
  - Higher level of abstraction
  - System (optionally) selects defaults

Cryptography

- Symmetric (e.g., 3DES) and Asymmetric (Public Key) cyphers
  - System provides all the algorithms currently supported by CryptoAPI (DES, 3DES, RC2, RSA, DSA, MD5, SHA1, …)
- Designed for extensibility and customization
  - Glue classes allow third parties to securely define crypto libraries
  - Allows developers to extend and

Sample: Encrypt Data

```
byte[] Encrypt(byte[] infoToEncrypt,
   byte[] key, byte[] initVector)
{
    DES_CSP des = new DES_CSP();
    SymmetricStreamEncryptor encryptor =
        des.CreateEncryptor(key,
   initVector);
    CryptoMemoryStream cms =
           new CryptoMemoryStream();
    encryptor.SetSink(cms);
    encryptor.Write(infoToEncrypt);
    encryptor.CloseStream();
    return cms.Data;
}
```

Questions?

Related Sessions

- 3-321: *Building International Applications for the .NET Framework, Part 1: Resource Model*
- 3-334: *.NET Framework Security*
- 3-326: *Writing Self-Describing Applications on the .NET Framework*
- 3-327: *Getting the Most out of the .NET Framework: Practices and Patterns*
- 3-332: *.NET Framework Performance Considerations*

Where do you want to go today?

Microsoft